1. **Write a Java Program to find GCD of two given numbers.**

public class RecursionGcd {

public static int gcd (int n1, int n2){

if (n2 == 0){

return n1;

}

return gcd(n2, n1 % n2);

}

public static void main (String [] args){

int n1 = 5;

int n2 = 8;

int gcd = gcd(n1, n2);

System.out.println("GCD OF "+ n1 +" AND " + n2 + " IS " + gcd);

}

}

OutPut:- GCD OF 5 AND 8 IS 1

1. **Write a java program to LCM of TWO given number.**

public class RecursionLcm{

static int hcf(int a, int b){

return b == 0 ? a : hcf(b, a % b);

}

public static void main (String [] args){

int n1 = 44;

int n2 = 22;

int hcf = hcf(n1,n2);

//System.out.println("The HCF is " + hcf);

int lcm = (n1 \* n2)/ hcf;

//

System.out.println("The Lcm is " + lcm );

}

}

Output:- The Lcm is 44

1. **Write a Java Program to print all the Prime Factorsof the Given Number.**

public class RecursionPrimeFactor{

static void factors(int n, int i){

if (i <= n){

if (n % i == 0){

System.out.println(i + " ");

}

factors(n, i + 1);

}

}

public static void main(String [] args)

{

int N = 16;

factors(N, 1);

}

}

Output:-

1

2

4

8

16

1. **Check whether the Given Numberis a Palindrome or NOT.**

public class RecursionPalindrome{

static int palindrome(int n, int temp){

if (n == 0)

return temp;

temp = (temp \* 10) + (n % 10);

return palindrome(n / 10, temp);

}

public static void main (String [] args){

int n = 232;

int temp = palindrome(n, 0);

if (temp == n)

System.out.println("Yes");

else

System.out.println("No");

}

}

Output:- Yes

1. **Write a Java Program to check whether the Given Number is Prime Number or NOT.**

public class RecursionPrimeNo{

public static boolean isPrime(int n, int i){

if ( n <= 2 )

return (n == 2) ? true : false;

if (n % i == 0)

return false;

if (i \* i > n)

return true;

return isPrime(n, i + 1);

}

public static void main (String [] args){

int n = 12;

if (isPrime(n, 2))

System.out.println("Prime Number");

else

System.out.println("Not Prime Number");

}

}

Output:-Not Prime Number

1. **Write a Java Program to check whether the given number is Armstrong Numberor NOT.**

**import java.util.Scanner;**

public class ArmstrongNumber {

// Function to calculate the power of a number

static int power(int n, int r) {

if (r == 0)

return 1;

else

return n \* power(n, r - 1);

}

// Function to count the number of digits in a number

static int countDigits(int num) {

if (num == 0)

return 0;

else

return 1 + countDigits(num / 10);

}

// Function to check if a number is an Armstrong number

static boolean isArmstrong(int num, int n) {

int sum = 0;

int temp = num;

int numberOfDigits = countDigits(num);

while (temp != 0) {

int digit = temp % 10;

sum += power(digit, numberOfDigits);

temp /= 10;

}

return sum == num;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a number: ");

int number = scanner.nextInt();

if (isArmstrong(number, countDigits(number)))

System.out.println(number + " is an Armstrong number.");

else

System.out.println(number + " is not an Armstrong number.");

scanner.close();

}

}

Output:-Enter a number:153

153 is a an Armstrong number.

1. **Write a Java Program to check whether the given number is Perfect Numberor NOT.**

import java.util.Scanner;

public class PerfectNumber {

// Function to calculate the sum of proper divisors of a number

static int sumOfDivisors(int n, int i) {

if (i == 1)

return 1;

else if (n % i == 0)

return i + sumOfDivisors(n, i - 1);

else

return sumOfDivisors(n, i - 1);

}

// Function to check if a number is a perfect number

static boolean isPerfect(int n) {

return sumOfDivisors(n, n / 2) == n;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a number: ");

int number = scanner.nextInt();

if (isPerfect(number))

System.out.println(number + " is a Perfect number.");

else

System.out.println(number + " is not a Perfect number.");

scanner.close();

    }

}

Output:- Enter a number: 58

58 is not a Perfect number.

**8.Write a Java Program to check whether the given numbers are Amicable Numbersor NOT**

public class AmicableNumber {

public static void main(String[] args) {

int number1 = 220;

int number2 = 284;

if (areAmicable(number1, number2)) {

System.out.println(number1 + " and " + number2 + " are amicable numbers.");

} else {

System.out.println(number1 + " and " + number2 + " are not amicable numbers.");

}

}

public static boolean areAmicable(int number1, int number2) {

return (sumOfDivisors(number1, 1, 0) == number2) && (sumOfDivisors(number2, 1, 0) == number1);

}

public static int sumOfDivisors(int number, int divisor, int sum) {

if (divisor > number / 2) {

return sum;

}

if (number % divisor == 0) {

sum += divisor;

}

return sumOfDivisors(number, divisor + 1, sum);

    }

}

Output:- 220 and 284 are amicable numbers.

9.**Write a Java Program to check whether the given number is Ramanujam's Numberor NOT.**

import java.util.HashMap;

import java.util.Map;

public class RamanujansNumber {

static boolean isRamanujansNumber(int num) {

Map<Integer, Integer> cubeMap = new HashMap<>();

// Finding pairs of cube sums

for (int i = 1; i \* i \* i < num; i++) {

for (int j = i + 1; j \* j \* j < num; j++) {

int sum = i \* i \* i + j \* j \* j;

if (cubeMap.containsKey(sum)) {

// Found a Ramanujan's number

System.out.println("Ramanujan's Number found: " + sum);

System.out.println("Pairs: (" + cubeMap.get(sum) + "^3 + " + i + "^3) and (" + cubeMap.get(sum) + "^3 + " + j + "^3)");

return true;

} else {

cubeMap.put(sum, i);

}

}

}

// No Ramanujan's number found

return false;

}

public static void main(String[] args) {

int number = 1729; // Example number

if (isRamanujansNumber(number))

System.out.println(number + " is a Ramanujan's number.");

else

System.out.println(number + " is not a Ramanujan's number.");

    }

}

Output:- Ramanujan's Number found: 1729

Pairs: (1^3 + 9^3) and (1^3 + 10^3)

1729 is a Ramanujan's number.

10.**Write a Java Program check whether the given number is Automorphic Numberor NOT.**

import java.util.Scanner;

public class AutomorphicNumber {

// Function to calculate the number of digits in a number using recursion

static int countDigits(int num) {

if (num == 0)

return 0;

else

return 1 + countDigits(num / 10);

}

// Function to check if a number is an Automorphic number

static boolean isAutomorphic(int num) {

int square = num \* num;

// Count the number of digits in the original number

int numDigits = countDigits(num);

// Extract the last 'numDigits' digits from the square

int lastDigits = square % (int) Math.pow(10, numDigits);

// Check if the extracted digits match the original number

return lastDigits == num;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a number: ");

int number = scanner.nextInt();

if (isAutomorphic(number))

System.out.println(number + " is an Automorphic number.");

else

System.out.println(number + " is not an Automorphic number.");

scanner.close();

}

}

Output:- Enter a number: 87

87 is not an Automorphic number.